INF3704 - Software Engineering Glossary

abstract data type
A type that is defined by its operations rather than its representation. The representation is private and may only be accessed by the defined operations.

acceptance testing
Customer tests of a system to decide if it is adequate to meet their needs and so should be accepted from a supplier.

activity chart
A chart used by project managers to show the dependencies between tasks that have to be completed. The chart shows the tasks, the time expected to complete these tasks and the task dependencies. The critical path is the longest path (in terms of the time required to complete the tasks) through the activity chart. The critical path defines the minimum time required to complete the project. Sometimes called a PERT chart.

Ada
A programming language that was developed for the US Department of Defense in the 1980s as a standard language for developing military software. It is based on programming language research from the 1970s and includes constructs such as abstract data types and support for concurrency. It is still used for large, complex military and aerospace systems.

agile manifesto
A set of principles encapsulating the ideas underlying agile methods of software development.

agile methods
Methods of software development that are geared to rapid software delivery. The software is developed and delivered in increments, and process documentation and
bureaucracy are minimized. The focus of development is on the code itself, rather than supporting documents.

algorithmic cost modeling
An approach to software cost estimation where a formula is used to estimate the project cost. The parameters in the formula are attributes of the project and the software itself.

application family
A set of software application programs that have a common architecture and generic functionality. These can be tailored to the needs of specific customers by modifying components and program parameters.

application framework
A set of reusable concrete and abstract classes that implement features common to many applications in a domain (e.g. user interfaces). The classes in the application framework are specialized and instantiated to create an application.

application program interface (API)
An interface, generally specified as a set of operations, that allows access to an application program's functionality. This means that this functionality can be called on directly by other programs and not just accessed through the user interface.

architectural pattern (style)
An abstract description of a software architecture that has been tried and tested in a number of different software systems. The pattern description includes information about where it is appropriate to use the pattern and the organization of the compo- nents of the architecture.

architectural view
A description of a software architecture from a particular perspective.

availability
The readiness of a system to deliver services when requested. Availability is usu- ally expressed as a decimal number, so an availability of 0.999 means that the sys- tem can deliver services for 999 out of 1000 time units.

B
A formal method of software development that is based on implementing a system by systematic transformation of a formal system specification.

bar chart (Gantt chart)
A chart used by project managers to show the project tasks, the schedule associated with these tasks and the people who will work on them. It shows the tasks' start and end dates and the staff allocations against a timeline.

black-box testing
An approach to testing where the testers have no access to the source code of a system or its components. The tests are derived from the system specification.

BPMN
Business Process Modeling Notation. A notation for defining workflows that describe business processes and service composition.

brownfield software development
The development of software for an environment where there are several existing systems that the software being developed must integrate with.

C
A programming language that was originally developed to implement the Unix sys- tem. C is a relatively low-level system implementation language that allows access to the system hardware and which can be compiled to efficient code. It is widely used for low-level systems programming and embedded systems development.

C++
An object-oriented programming language that is a superset of C.

C#
An object-oriented programming language, developed by Microsoft, that has much in common with C++, but which includes features that allow more compile-time type checking.

Capability Maturity Model (CMM)
The Software Engineering Institute's Capability Maturity Model, which is used to assess the level of software development maturity in an organization. It has now been superseded by CMMI, but is still widely used.

Computer-Aided Software Engineering (CASE)
The term that was invented in the 1980s to describe process of developing software using automated tool support. Virtually all software development is now reliant on tool support so the term 'CASE is no longer widely used.

CASE tool
A software tool, such as a design editor or a program debugger, used to support an activity in the software development process.

CASE workbench
An integrated set of CASE tools that work together to support a major process activity such as software design or configuration management. Now often called a programming environment.

change management
A process to record, check, analyze, estimate and implement proposed changes to a software system.

class diagram
A UML diagram types that shows the object classes in a system and their relationships.

client–server architecture
An architectural model for distributed systems where the system functionality is offered as a set of services provided by a server. These are accessed by client com- puters that make use of the services. Variants of this approach, such as three-tier client–server architectures, use multiple servers.

cloud computing
The provision of computing and/or application services over the Internet using a 'cloud' of servers from an external provider. The 'cloud' is implemented using a large number of commodity computers and virtualization technology to make effective use of these systems.

CMMI
An integrated approach to process capability maturity modeling based on the adop- tion of good software engineering practice and integrated quality management. It supports discrete and continuous maturity modeling and integrates systems and software engineering process maturity models. Developed from the original Capa- bility Maturity Model.

COCOMO II
See Constructive Cost Modeling.

code of ethics and professional practice
A set of guidelines that set out expected ethical and professional behavior for software engineers. This was defined by the major US professional societies (the ACM and the IEEE) and defines ethical behavior under eight headings: public, client and employer, product, judgment, management, colleagues, profession and self.

Common Request Broker Architecture (CORBA)
A set of standards proposed by the Object Management Group (OMG) that defines distributed component models and communications. Influential in the development of distributed systems but no longer widely used.

component
A deployable, independent unit of software that is completely defined and accessed through a set of interfaces.

component model
A set of standards for component implementation, documentation and deploy- ment. These cover the specific interfaces that may be provided by a compo- nent, component naming, component interoperation and component composition. Component models provide the basis for middleware to support executing components.

component-based software engineering (CBSE)
The development of software by composing independent, deployable software components that are consistent with a component model.

conceptual design
The development of a high-level vision of a complex system and a description of its essential capabilities. Designed to be understood by people who are not systems engineers.

configurable application system
An application system product, developed by a system vendor, that offers function- ality that may be configured for use in different companies and environments.

configuration item
A machine-readable unit, such as a document or a source code file, that is subject to change and where the change has to be controlled by a configuration management system.

configuration management
The process of managing the changes to an evolving software product. Configura- tion management involves version management, system building, change manage- ment and release management.

Constructive Cost Modeling (COCOMO)
A family of algorithmic cost estimation models. COCOMO was first proposed in the early-1980s and has been modified and updated since then to reflect new tech- nology and changing software engineering practice. COCOMO II is its latest instantiation and is a freely available algorithmic cost estimation model that is supported by open source software tools.

CORBA
See Common Request Broker Architecture.

control metric
A software metric that allows managers to make planning decisions based on infor- mation about the software process or the software product that is being developed. Most control metrics are process metrics.

critical system
A computer system whose failure can result in significant economic, human or environmental losses.

COTS system
A Commercial Off-the-Shelf system. The term COTS is now mostly used in military systems. See configurable application system.

CVS
A widely used, open-source software tool used for version management.

data processing system
A system that aims to process large amounts of structured data. These systems usually process the data in batches and follow an input-process- output model. Examples of data processing systems are billing and invoicing systems, and payment systems.

denial of service attack
An attack on a web-based software system that attempts to overload the system so that it cannot provide its normal service to users.

dependability
The dependability of a system is an aggregate property that takes into account the system’s safety, reliability, availability, security, resilience and other attributes. The dependability of a system reflects the extent to which it can be trusted by its users.

dependability requirement
A system requirement that is included to help achieve the required dependability for a system. Non-functional dependability requirements specify dependability attribute values; functional dependability requirements are functional requirements that specify how to avoid, detect, tolerate or recover from system faults and failures.

dependability case
A structured document that is used to back up claims made by a system developer about the dependability of a system. Specific types of dependability case are safety cases and security cases.

design pattern
A well-tried solution to a common problem that captures experience and good prac- tice in a form that can be reused. It is an abstract representation than can be instantiated in a number of ways.

digital learning environment
An integrated set of software tools, educational applications and content that is geared to support learning.

distributed system
A software system where the software sub-systems or components execute on different processors.

domain
A specific problem or business area where software systems are used. Examples of domains include real-time control, business data processing and telecommunica- tions switching.

domain model
A definition of domain abstractions, such as policies, procedures, objects, relationships and events. It serves as a base of knowledge about some problem area.

DSDM
Dynamic System Development Method. Claimed to be one of the first agile devel- opment methods.

embedded system
A software system that is embedded in a hardware device e.g. the software system in a cell phone. Embedded systems are usually real-time systems and so have to respond in a timely way to events occurring in their environment.

emergent property
A property that only becomes apparent once all of the components of the system have been integrated to create the system.

Enterprise Java Beans (EJB)
A Java-based component model.

enterprise resource planning (ERP) system
A large-scale software system that includes a range of capabilities to support the operation of business enterprises and which provides a means of sharing informa- tion across these capabilities. For example, an ERP system may include support for supply chain management, manufacturing and distribution. ERP systems are con- figured to the requirements of each company using the system.

ethnography
An observational technique that may be used in requirements elicitation and analy- sis. The ethnographer immerses him or herself in the users' environment and observes their day-to-day work habits. Requirements for software support can be inferred from these observations.

event-based systems
Systems where the control of operation is determined by events that are generated in the system's environment. Most real-time systems are event-based systems.

extreme programming (XP)
A widely-used agile method of software development that includes practices such as scenario-based requirements, test-first development and pair programming.

fault avoidance
Developing software in such a way that faults are not introduced into that software.

fault detection
The use of processes and run-time checking to detect and remove faults in a program before these result in a system failure.

fault tolerance
The ability of a system to continue in execution even after faults have occurred.

fault-tolerant architectures
System architectures that are designed to allow recovery from software faults. These are based on redundant and diverse software components.

formal methods
Methods of software development where the software is modeled using formal math- ematical constructs such as predicates and sets. Formal transformation converts this model to code. Mostly used in the specification and development of critical systems.

Gantt chart
See bar chart.

Git
A distributed version management and system building tool where developers take complete copies of the project repository to allow concurrent working.

GitHub
A server that maintains a large number of Git repositories. Repositories may be private or public. The repositories for many open-source projects are maintained on GitHub.

hazard
A condition or state in a system that has the potential to cause or contribute to an accident.

host-target development
A mode of software development where the software is developed on a separate computer from where it is executed. The normal approach to development for embedded and mobile systems.

iLearn system
A digital learning environment to support learning in schools. Used as a case study in this book.

incremental development
An approach to software development where the software is delivered and deployed in increments.

information hiding
Using programming language constructs to conceal the representation of data struc- tures and to control external access to these structures.

inspection
See program inspection.

insulin pump
A software-controlled medical device that can deliver controlled doses of insulin to people suffering from diabetes. Used as a case study in this book.

integrated application system
An application system that is created by integrating two or more configurable application systems or legacy systems.

interface
A specification of the attributes and operations associated with a software compo- nent. The interface is used as the means of accessing the component's functionality.

ISO 9000/9001
A set of standards for quality management processes that is defined by the Interna- tional Standards Organization (ISO). ISO 9001 is the ISO standard that is most applicable to software development. These may be used to certify the quality management processes in an organization.

iterative development
An approach to software development where the processes of specification, design, programming and testing are interleaved.

J2EE
Java 2 Platform Enterprise Edition. A complex middleware system that supports the development of component-based web applications in Java. It includes a component model for Java components, APIs, services, etc.

Java
A widely used object-oriented programming language that was designed by Sun (now Oracle) with the aim of platform independence.

language processing system
A system that translates one language into another. For example, a compiler is a language-processing system that translates program source code to object code.

legacy system
A socio-technical system that is useful or essential to an organization but which has been developed using obsolete technology or methods. Because legacy systems often perform critical business functions, they have to be maintained.

Lehman's Laws
A set of hypotheses about the factors that influence the evolution of complex software systems.

maintenance
The process of making changes to a system after it has been put into operation.

mean time to failure (MTTF)
The average time between observed system failures. Used in reliability specification.

Mentcare system
Mental Health Care Patient Management System. This is a system used to record information about consultations and treatments prescribed for people suffering from mental health problems. Used as a case study in this book.

middleware
The infrastructure software in a distributed system. It helps manage interactions between the distributed entities in the system and the system databases. Examples of middleware are an object request broker and a transaction management system.

misuse case
A description of a possible attack on a system that is associated with a system use case.

model-driven architecture (MDA)
An approach to software development based on the construction of a set of system models, which can be automatically or semi-automatically processed to generate an executable system.

model checking
A method of static verification where a state model of a system is exhaustively ana- lyzed in an attempt to discover unreachable states.

model-driven development (MDD)
An approach to software engineering centered around system models that are expressed in the UML, rather than programming language code. This extends MDA to consider activities other than development such as requirements engineering and testing.

multi-tenant databases
Databases where information from several different organizations is stored in the same database. Used in the implementation of software as a service.

mutual exclusion
A mechanism to ensure that a concurrent process maintains control of memory until updates or accesses have been completed.

.NET
A very extensive framework used to develop applications for Microsoft Windows systems. Includes a component model that defines standards for components in Windows systems and associated middleware to support component execution.

object class
An object class defines the attributes and operations of objects. Objects are created at run-time by instantiating the class definition. The object class name can be used as a type name in some object-oriented languages.

object model
A model of a software system that is structured and organized as a set of object classes and the relationships between these classes. Various different perspectives on the model may exist such as a state perspective and a sequence perspective.

object-oriented (OO) development
An approach to software development where the fundamental abstractions in the system are independent objects. The same type of abstraction is used during specification, design and development.

object constraint language (OCL)
A language that is part of the UML, used to define predicates that apply to object classes and interactions in a UML model. The use of the OCL to specify compo- nents is a fundamental part of model-driven development.

Object Management Group (OMG)
A group of companies formed to develop standards for object-oriented develop- ment. Examples of standards promoted by the OMG are CORBA, UML and MDA.

open source
An approach to software development where the source code for a system is made public and external users are encouraged to participate in the development of the system.

operational profile
A set of artificial system inputs that reflect the pattern of inputs that are processed in an operational system. Used in reliability testing.

pair programming
A development situation where programmers work in pairs, rather than individually, to develop code. A fundamental part of extreme programming.

peer-to-peer system
A distributed system where there is no distinction between clients and servers. Computers in the system can act as both clients and servers. Peer-to-peer applications include file sharing, instant messaging and cooperation support systems.

People Capability Maturity Model (P-CMM)
A process maturity model that reflects how effective an organization is at managing the skills, training and experience of the people in that organization.

plan-driven process
A software process where all of the process activities are planned before the soft- ware is developed.

planning game
An approach to project planning based on estimating the time required to imple- ment user stories. Used in some agile methods.

predictor metric
A software metric that is used as a basis for making predictions about the characteristics of a software system, such as its reliability or maintainability.

probability of failure on demand (POFOD)
A reliability metric that is based on the likelihood of a software system failing when a demand for its services is made.

process improvement
Changing a software development process with the aim of making that process more efficient or improving the quality of its outputs. For example, if your aim is to reduce the number of defects in the delivered software, you might improve a process by adding new validation activities.

process model
An abstract representation of a process. Process models may be developed from various perspectives and can show the activities involved in a process, the artifacts used in the process, constraints that apply to the process, and the roles of the people enacting the process.

process maturity model
A model of the extent to which a process includes good practice and reflective and measurement capabilities that are geared to process improvement.

program evolution dynamics
The study of the ways in which an evolving software system changes. It is claimed that Lehman’s Laws govern the dynamics of program evolution.

program generator
A program that generates another program from a high-level, abstract specification. The generator embeds knowledge that is reused in each generation activity.

program inspection
A review where a group of inspectors examine a program, line by line, with the aim of detecting program errors. A checklist of common programming errors often drives inspections.

Python
A programming language with dynamic types, which is particularly well- suited to the development of web-based systems.

quality management (QM)
The set of processes concerned with defining how software quality can be achieved and how the organization developing the software knows that the software has met the required level of quality.

quality plan
A plan that defines the quality processes and procedures that should be used. This involves selecting and instantiating standards for products and processes and defin- ing the system quality attributes that are most important.

rapid application development (RAD)
An approach to software development aimed at rapid delivery of the software. It often involves the use of database programming and development support tools such as screen and report generators.

rate of occurrence of failure (ROCOF)
A reliability metric that is based on the number of observed failures of a system in a given time period.

Rational Unified Process (RUP)
A generic software process model that presents software development as a four- phase iterative activity, where the phases are inception, elaboration, construction and transition. Inception establishes a business case for the system, elaboration defines the architecture, construction implements the system, and transition deploys the system in the customer's environment.

real-time system
A system that has to recognize and process external events in 'real-time'. The correctness of the system does not just depend on what it does but also on how quickly it does it. Real-time systems are usually organized as a set of concurrent processes.

reductionism
An engineering approach that relies on breaking down a problem to sub- problems, solving these sub-problems independently then integrating these solutions to create the solution to the larger problem.

reengineering
The modification of a software system to make it easier to understand and change. Reengineering often involves software and data restructuring and organization, program simplification and redocumentation.

reengineering, business process
Changing a business process to meet a new organizational objective such as reduced cost and faster execution.

refactoring
Modifying a program to improve its structure and readability without changing its functionality.

reference architecture
A generic, idealized architecture that includes all the features that systems might incorporate. It is a way of informing designers about the general structure of that class of system rather than a basis for creating a specific system architecture.

release
A version of a software system that is made available to system customers.

reliability
The ability of a system to deliver services as specified. Reliability can be specified quantitatively as a probability of failure on demand or as the rate of occurrence of failure.

reliability growth modeling
The development of a model of how the reliability of a system changes (improves) as it is tested and program defects are removed.

requirement, functional
A statement of some function or feature that should be implemented in a system.

requirement, non-functional
A statement of a constraint or expected behavior that applies to a system. This constraint may refer to the emergent properties of the software that is being developed or to the development process.

requirements management
The process of managing changes to requirements to ensure that the changes made are properly analyzed and tracked through the system.

resilience
A judgement of how well a system can maintain the continuity of its critical services in the presence of disruptive events, such as equipment failure and cyberattacks.

REST
REST (Representational State Transfer) is a style of development based around simple client/server interaction which uses the HTTP protocol for communications. REST is based around the idea of an identifiable resource, which has a URI. All interaction with resources is based on HTTP POST, GET, PUT and DELETE. Widely used for implementing low overhead web services (RESTful services).

revision control systems
See version control systems.

risk
An undesirable outcome that poses a threat to the achievement of some objective. A process risk threatens the schedule or cost of a process; a product risk is a risk that may mean that some of the system requirements may not be achieved. A safety risk is a measure of the probability that a hazard will lead to an accident.

risk management
The process of identifying risks, assessing their severity, planning measures to put in place if the risks arise and monitoring the software and the software process for risks.

Ruby
A programming language with dynamic types that is particularly well- suited to web application programming.

SaaS
See software as a service.

safety
The ability of a system to operate without behavior that may injure or kill people or damage the system’s environment.

safety case
A body of evidence and structured argument from that evidence that a system is safe and/or secure. Many critical systems must have associated safety cases that are assessed and approved by external regulators before the system is certified for use.

SAP
A German company that has developed a well-known and widely-used ERP system. It also refers to the name given to the ERP system itself.

scenario
A description of one typical way in which a system is used or a user carries out some activity.

scenario testing
An approach to software testing where test cases are derived from a scenario of system use.

Scrum
An agile method of development, which is based on sprints – short development, cycles. Scrum may be used as a basis for agile project management alongside other agile methods such as XP.

security
The ability of a system to protect itself against accidental or deliberate intrusion. Security includes confidentiality, integrity and availability.

SEI
Software Engineering Institute. A software engineering research and technology transfer center, founded with the aim of improving the standard of software engineering in US companies.

sequence diagram
A diagram that shows the sequence of interactions required to complete some operation. In the UML, sequence diagrams may be associated with use cases.

server
A program that provides a service to other (client) programs.

service
See web service.

socio-technical system
A system, including hardware and software components, that has defined operational processes followed by human operators and which operates within an organization. It is therefore influenced by organizational policies, procedures and structures.

software analytics
Automated analysis of static and dynamic data about software systems to discover relationships between these data. These relationships may provide insights about possible ways to improve the quality of the software.

software architecture
A model of the fundamental structure and organization of a software system.

software as a service (SaaS)
Software applications that are accessed remotely through a web browser rather than installed on local computers. Increasingly used to deliver application services to end-users.

software development life cycle
Often used as another name for the software process. Originally coined to refer to the waterfall model of the software process.

software metric
An attribute of a software system or process that can be expressed numerically and measured. Process metrics are attributes of the process such as the time taken to complete a task; product metrics are attributes of the software itself such as size or complexity.

software process
The activities and processes that are involved in developing and evolving a soft- ware system.

software product line
See application family.

spiral model
A model of a development process where the process is represented as a spiral, with each round of the spiral incorporating the different stages in the process. As you move from one round of the spiral to another, you repeat all of the stages of the process.

state diagram
A UML diagram type that shows the states of a system and the events that trigger a transition from one state to another.

static analysis
Tool-based analysis of a program’s source code to discover errors and anomalies. Anomalies, such as successive assignments to a variable with no intermediate use may be indicators of programming errors.

structured method
A method of software design that defines the system models that should be devel- oped, the rules and guidelines that should apply to these models and a process to be followed in developing the design.

Structured Query Language (SQL)
A standard language used for relational database programming.

Subversion
A widely-used, open source version control and system building tool that is avail- able on a range of platforms.

Swiss cheese model
A model of system defenses against operator failure or cyberattack that takes vul- nerabilities in these

defenses into account.

system
A system is a purposeful collection of interrelated components, of different kinds, which work together to deliver a set of services to the system owner and users.

system building
The process of compiling the components or units that make up a system and link- ing these with other components to create an executable program. System building is normally automated so that recompilation is minimized. This automation may be built in to the language processing system (as in Java) or may involve software tools to support system building.

systems engineering
A process that is concerned with specifying a system, integrating its components and testing that the system meets its requirements. System engineering is concerned with the whole socio-technical system—software, hardware and operational processes—not just the system software.

system of systems
A system that is created by integrating two or more existing systems.

system testing
The testing of a completed system before it is delivered to customers.

test coverage
The effectiveness of system tests in testing the code of an entire system. Some companies have standards for test coverage e.g. the system tests shall ensure that all program statements are executed at least once.

test-driven development
An approach to software development where executable tests are written before the program code. The set of tests are run automatically after every change to the program.

TOGAF
An architectural framework, supported by the Object Management Group, that is intended to support the development of enterprise architectures for systems of systems.

transaction
A unit of interaction with a computer system. Transactions are independent and atomic (they are not broken down into smaller units) and are a fundamental unit of recovery, consistency and concurrency.

transaction processing system
A system that ensures that transactions are processed in such a way so that they do not interfere with each other and so that individual transaction failure does not affect other transactions or the system’s data.

Unified Modeling Language (UML)
A graphical language used in object-oriented development that includes several types of system model that provide different views of a system. The UML has become a *de facto* standard for object-oriented modeling.

unit testing
The testing of individual program units by the software developer or development team.

use case
A specification of one type of interaction with a system.

use-case diagram
A UML diagram type that is used to identify use-cases and graphically depict the users involved. It must be supplemented with additional information to completely describe use-cases.

user interface design
The process of designing the way in which system users can access system functionality, and the way that information produced by the system is displayed.

user story
A natural language description of a situation that explains how a system or systems might be used and the interactions with the systems that might take place.

validation
The process of checking that a system meets the needs and expectations of the customer.

verification
The process of checking that a system meets its specification.

version control
The process of managing changes to a software system and its components so that it is possible to know which changes have been implemented in each version of the component/system, and also to recover/recreate previous versions of the component/system.

version control (VC) systems
Software tools that have been developed to support the processes of version control. These may be based on either centralized or distributed repositories.

waterfall model
A software process model that involves discrete development stages: specification, design, implementation, testing and maintenance. In principle, one stage must be complete before progress to the next stage is possible. In practice, there is significant iteration between stages.

web service
An independent software component that can be accessed through the Internet using standard protocols. It is completely self-contained without external dependencies. XML-based standards such as SOAP (Standard Object Access Protocol), for web service information exchange, and WSDL (Web Service Definition Language), for the definition of web service interfaces, have been developed. However, the REST approach may also be used for web service implementation.

white-box testing
An approach to program testing where the tests are based on knowledge of the structure of the program and its components. Access to source code is essential for white-box testing.

wicked problem
A problem that cannot be completely specified or understood because of the complexity of the interactions between the elements that contribute to the problem.

wilderness weather system
A system to collect data about the weather conditions in remote areas. Used as a case study in this book.

workflow
A detailed definition of a business process that is intended to accomplish a certain task. The workflow is usually expressed graphically and shows the individual process activities and the information that is produced and consumed by each activity.

WSDL
An XML-based notation for defining the interface of web services.

XML
Extended Markup Language. XML is a text markup language that supports the interchange of structured data. Each data field is delimited by tags that give information about that field. XML is now very widely used and has become the basis of protocols for web services.

XP
See Extreme Programming.

Z
A model-based, formal specification language developed at the University of Oxford in England.